AWS
Black Belt
Online Seminar

【AWS Black Belt Online Seminar】
AWS Key Management Service (AWS KMS)

アマゾンウェブサービスジャパン株式会社
ソリューションアーキテクト　布目　拓也
2016.09.28

amazon web services

内容についての注意点

- **本資料では2016年9月28日時点のサービス内容および価格についてご説明しています。最新の情報はAWS公式ウェブサイト(http://aws.amazon.com)にてご確認ください。**

- **資料作成には十分注意しておりますが、資料内の価格とAWS公式ウェブサイト記載の価格に相違があった場合、AWS公式ウェブサイトの価格を優先とさせていただきます。**

- **価格は税抜表記となっています。日本居住者のお客様が東京リージョンを使用する場合、別途消費税をご請求させていただきます。**

AWS does not offer binding price quotes. AWS pricing is publicly available and is subject to change in accordance with the AWS Customer Agreement available at http://aws.amazon.com/agreement/. Any pricing information included in this document is provided only as an estimate of usage charges for AWS services based on certain information that you have provided. Monthly charges will be based on your actual use of AWS services, and may vary from the estimates provided.

アジェンダ

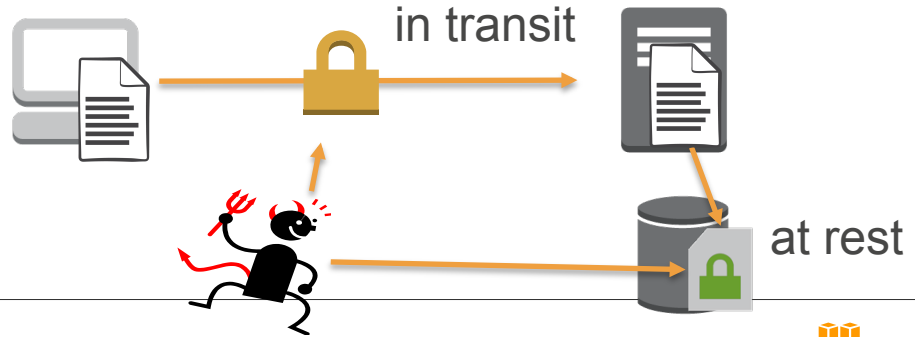
- KMSの概要
- KMSの暗号鍵管理
- KMSの暗号化処理
- KMS利用TIPS
- ログ、HSMとの比較、制限、料金等
- まとめ

アジェンダ

- KMSの概要
- KMSの暗号鍵管理
- KMSの暗号化処理
- KMS利用TIPS
- ログ、HSMとの比較、制限、料金等
- まとめ

暗号化を利用したデータ保護

- 通信の暗号化(Data encryption in transit)
  - 通信路の盗聴からデータを守る
    - SSL/TLS
    - IPSec
    - 無線LAN暗号化
- 保管データの暗号化(Data Encryption at rest)
  - 保管されたデータが意図しない第三者から読み出されるのを防ぐ
    - ファイル暗号化
    - 暗号化ファイルシステム
    - データベース暗号化
    - ブロックデバイス暗号化



暗号化の鍵管理において考慮すべき問題

- 鍵はどこに保管されるのか？
  - 自身が所有するハードウェア?
  - AWSのハードウェア？
- 鍵はどこで使われるのか？
  - 自分で管理するクライアントソフトウェア上？
  - AWSがコントロールを提供するサーバソフトウェア上？
- 誰が鍵を使えるのか？
  - 許可をもつユーザーやアプリケーション?
  - ユーザーが許可を与えたAWS上のアプリケーション？
- 鍵関連の**確かな**セキュリティを担保する仕組みがあるのか？

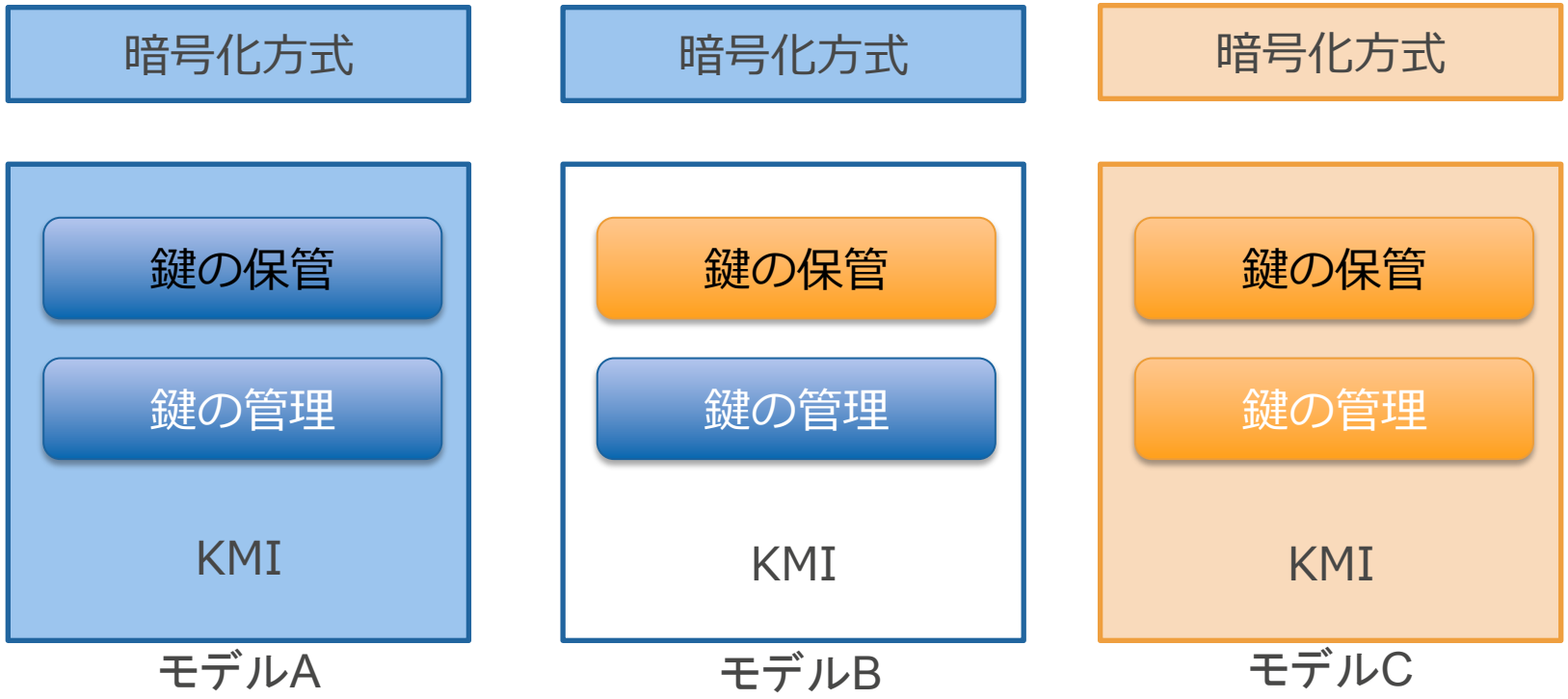
Key Management Infrastructure

- KMI(Key Management infrastructure) とは？
  - 暗号鍵の保管、鍵のアクセス制御等、鍵自身のセキュリティを管理するインフラストラクチャ

- KMIは２つのサブコンポーネントから成り立つ：
  - 平文の鍵を保護するためのストレージレイヤー
  - 鍵の利用を認可する管理レイヤー

- コンプライアンス要件が高い環境ではハードウェアセキュリティモジュール(HSM)が利用される事が多い
  - 鍵保管のための占有ストレージ、耐タンパー性、認可の無いユーザーからの保護

AWSにおける暗号化モデルとKMI



お客様が管理　AWSが管理

AWS Key Management Service (AWS KMS)

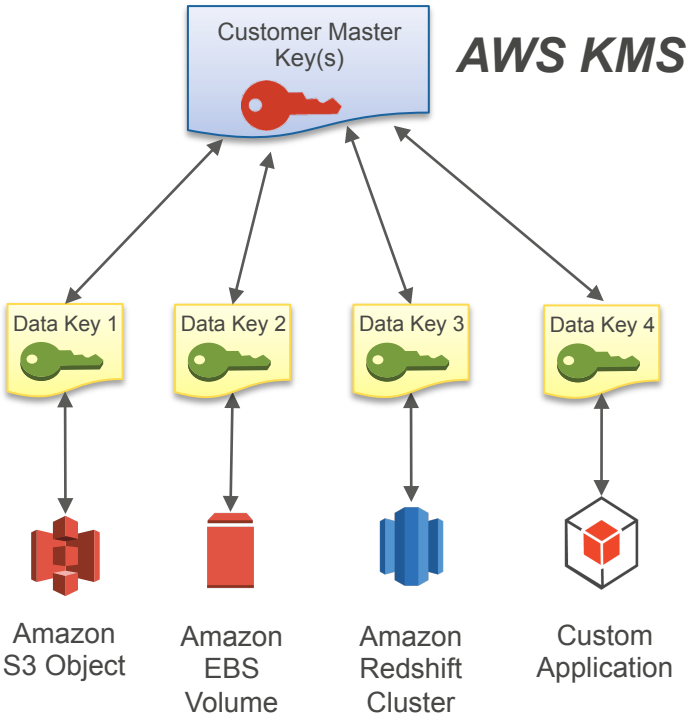
- 暗号鍵の作成、管理、運用サービス
  - 可用性、物理的セキュリティ、ハードウェアの管理をAWS が担当するマネージドサービス
  - 暗号化キーを保存および使用するための安全なロケーションを提供
    - 暗号鍵の可用性、機密性を確保
    - 低コストで使用可能
- S3, EBS, Redshift等のAWSサービスとの統合
- SDKとの連携でお客様の独自アプリケーションデータも暗号化
- AWS CloudTrail と連動したログの生成による組み込み型監査
- 中国リージョンを除く全てのリージョンで利用可能

KMSで使用する重要な用語

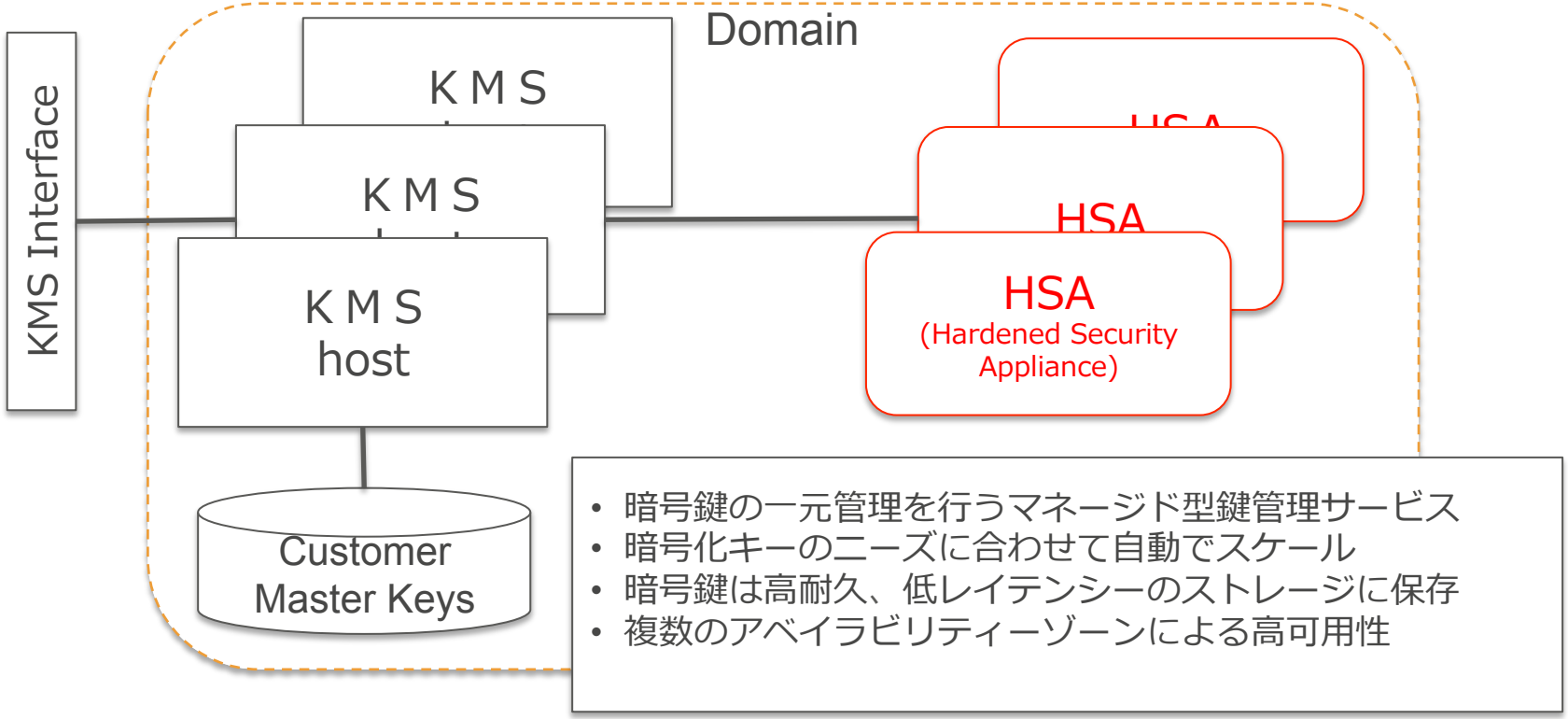
- Customer Master Key（CMK）
  - 暗号鍵のヒエラルキーの頂点に位置するKMS上の鍵
  - 外部にエクスポートされない
  - 暗号化された上で可用性の高いストレージに保管される
  - KMS内での直接暗号化/復号に利用される（ただし4KBまでのデータ）
- Customer Data Key (CDK)
  - CMKから生成される暗号鍵
  - 大量のデータの暗号化に使用
  - 暗号化されない形でもエクスポートが可能
- Envelope Encryption
  - マスターキーをデータ暗号化に直接利用するのではなく暗号化に利用するためのデータキーを暗号化/復号に利用
  - 暗号化したデータと、CMKで暗号化されたデータキーを一緒に保管することで鍵管理の複雑性を軽減
  - 暗号化されているデータキーをCMKで復号すれば、データ本体の復号も可能
  - ラージデータの暗号化に対応可能

KMSにおける暗号鍵のヒエラルキー

- 2-Tierの暗号鍵ヒエラルキー
- 個別のデータキーによるユーザーデータの暗号化
- AWS KMS マスターキーによるデータキーの暗号化
- Envelope Encryptionを利用
- Envelope Encryptionをの利点：
  - データキーの漏洩リスクを限定化
  - ラージデータを暗号化する場合のパフォーマンスメリット
  - 少数のマスターキーを管理することで管理性を向上
  - 鍵利用に関する中央集中アクセスと監査

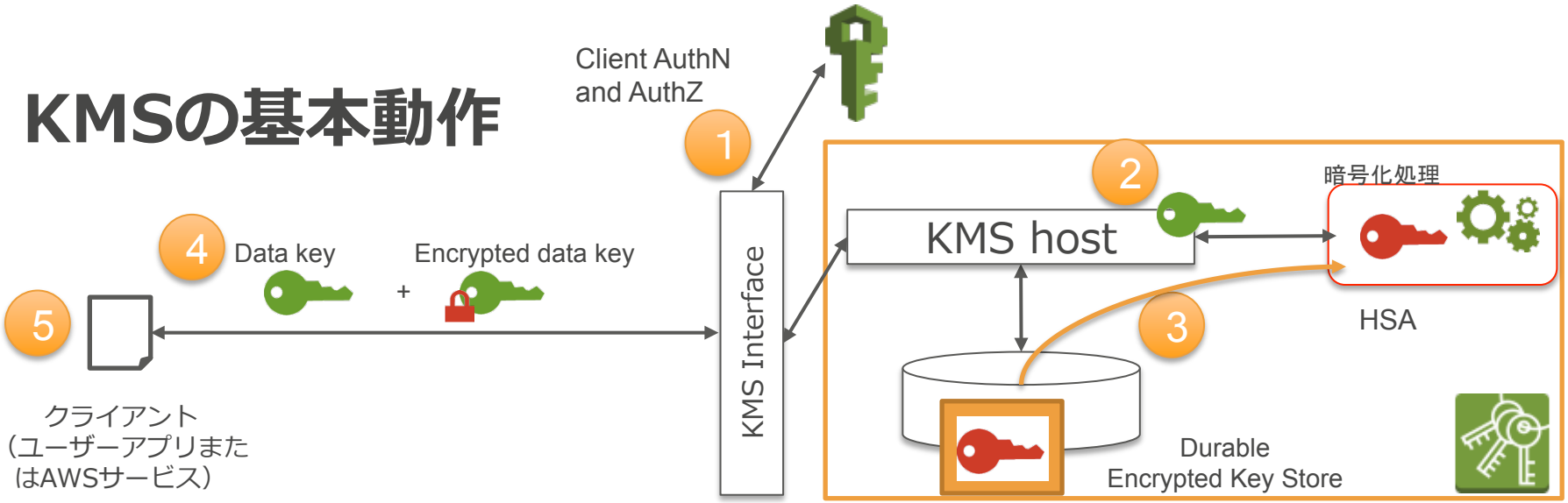


KMSのアーキテクチャー



- 暗号鍵の一元管理を行うマネージド型鍵管理サービス
- 暗号化キーのニーズに合わせて自動でスケール
- 暗号鍵は高耐久、低レイテンシーのストレージに保存
- 複数のアベイラビリティーゾーンによる高可用性

KMSの基本動作



1. クライアントは AWSアカウント内のKMSマスターキーのIDを渡して`kms:GenerateDataKey` をCallする。クライアントのリクエストは呼び出したユーザーの権限と鍵のアクセス権限の両方に基づき認証される
2. KMSがユニークなデータキーを生成
3. 暗号化されたカスタマースターキーを高耐久ストレージから取り出し、HSAの中で復号する
4. KMSはカスタマーマスターキーを使ってデータキーを暗号化し、平文のデータキーと暗号化されたデータキーがクライアントに返される
5. クライアントは、平文のデータキーを使ってデータを暗号化し、データとともに暗号化されたデータキーを保存する

復号：復号の際は、クライアントがデータに含まれる暗号化されたデータキーをKMSに渡す。暗号化されたデータキーは復号のために必要

KMSでできること/できないこと

できること

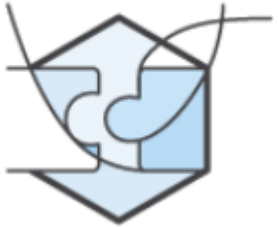
- 暗号鍵の生成と安全な保管
- 鍵利用の権限管理
- 鍵利用の監査
- 対称鍵暗号
- 4KBまでのデータ暗号化※
- AWSサービスとのインテグレーション

現時点でできないこと

- シングルテナント
- 非対称鍵暗号
- 4KBを超えるデータの直接的な暗号化
- 鍵のエクスポート

※鍵データの暗号化を対象

KMSとインテグレーションされているAWSサービス



様々なAWSサービスとのインテグレーションにより、容易に利用可能

カテゴリ	サービス
ストレージ・コンテンツ配信	Amazon S3, Amazon EBS, AWS Import/Export Snowball
データベース	Amazon RDS, Amazon Redshift, AWS Database Migration Service
開発者ツール	AWS CodeCommit※
分析	Amazon EMR, Amazon Kinesis Firehose
アプリケーションサービス	Amazon Elastic Transcoder, Amazon SES
エンタープライズアプリケーション	Amazon WorkSpaces, Amazon WorkMail

https://aws.amazon.com/jp/kms/details/

※2016/09 現在、CodeCommitはAWS管理キーのみサポート

主要なKMS API

- **鍵管理用API**
  - CreateKey, CreateAlias
  - DisableKey
  - EnableKeyRotation
  - PutKeyPolicy
  - ListKeys, DescribeKey
- **データAPI**
  - Encrypt
  - Decrypt
  - ReEncrypt
  - GenerateDataKey

26 API actions and growing

http://docs.aws.amazon.com/kms/latest/APIReference/Welcome.html

アジェンダ

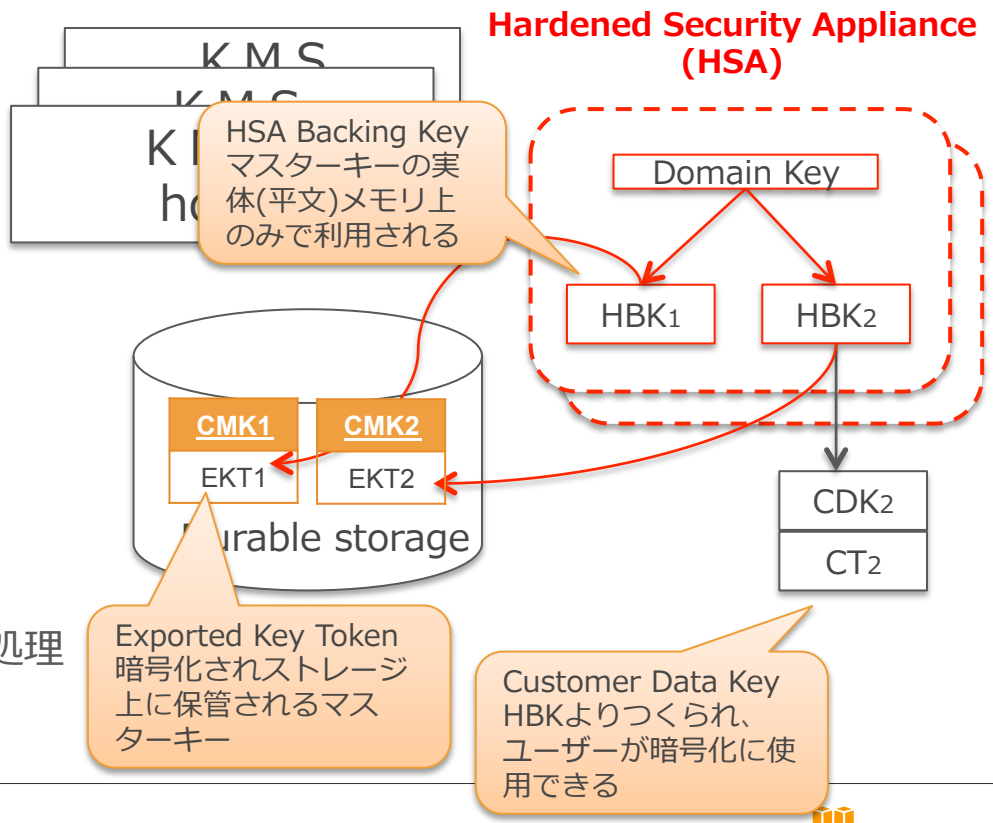
- KMSの概要
- KMSの鍵管理
- KMSの暗号化処理
- KMS利用TIPS
- ログ、HSMとの比較、制限、料金等
- まとめ

KMSの暗号鍵管理機能

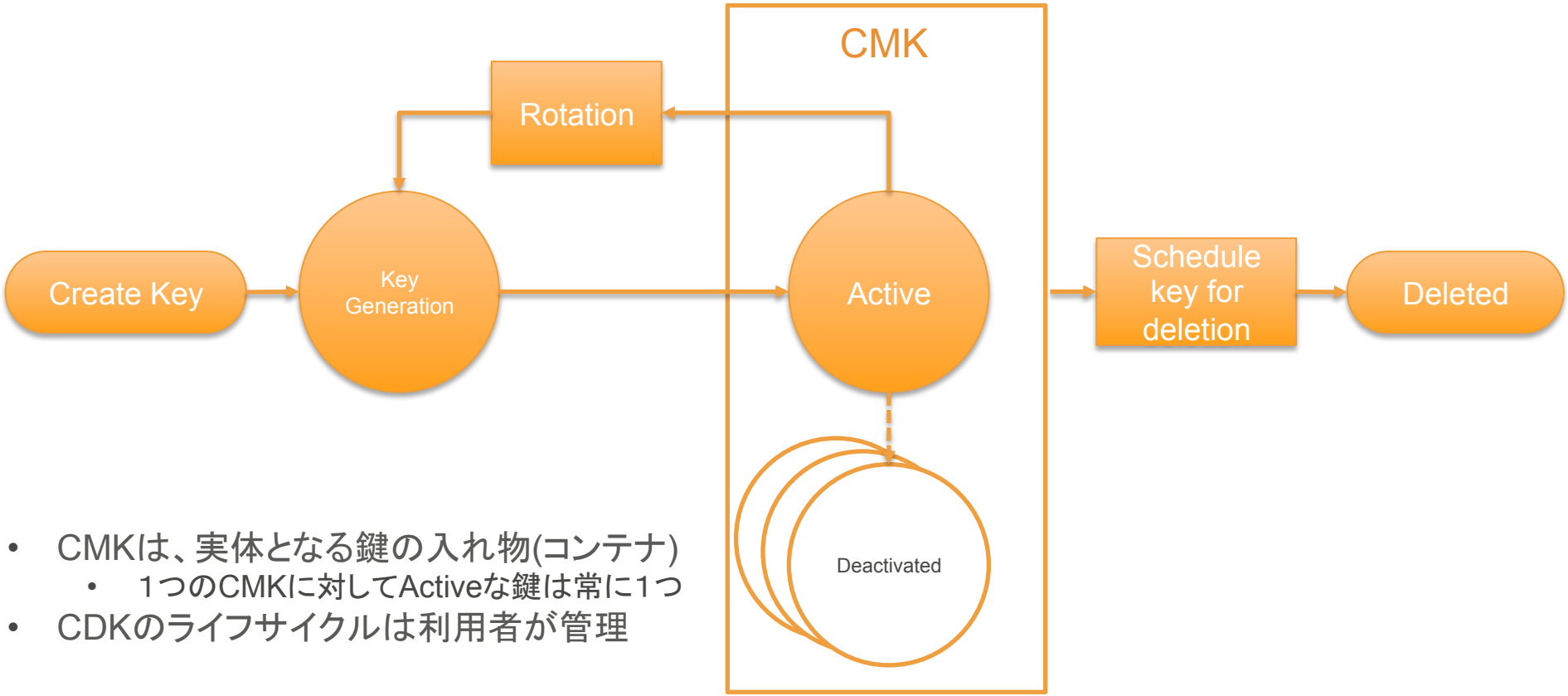
- 単一のエイリアスおよびDescriptionを付けたCMKの作成
- キーを管理するIAM ユーザーおよびロールの定義
- キーを使用できるIAM ユーザーおよびロールの定義
- CMKの無効化/有効化/**削除**
- 1 年ごとの自動キーローテーションの設定
- CMKの**インポート** New

KMSの鍵管理

- KMS内部では以下の鍵データが存在する
  - Domain Key(DK)
    - 管理用の鍵
    - リージョン内の全てのHSAが**メモリ上**に保持
    - 日次でローテーション
  - HSA Backing Key(HBK)
    - CMKの実体となる平文の鍵データ
    - CMK生成時にHSAにて生成される
    - **HSAのメモリ上にのみ存在**
    - 平文でのExport不可
  - Exported Key Token(EKT)
    - DKで暗号化されたHBK
    - KMS内のDurable Storageに保管される
    - 必要に応じてHSA上で復号化される
- CMKは論理キー(鍵のコンテナ)
  - HBK、EKTが関連付けられている
- ユーザーはCMKからCDKを生成して暗号化処理に利用

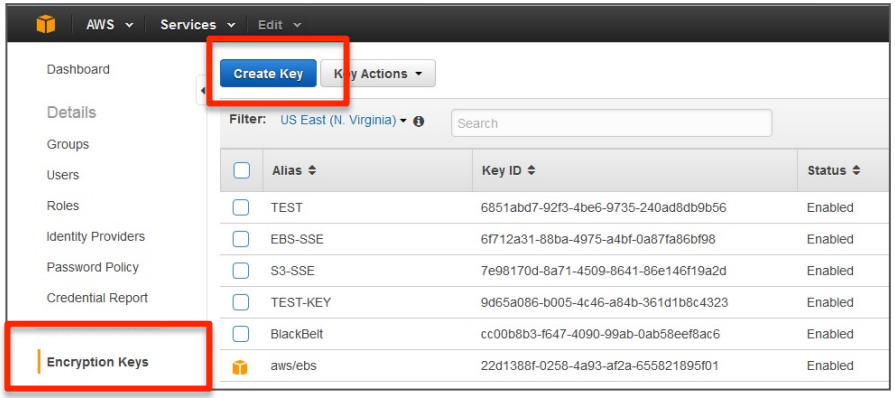


暗号鍵（CMK）のライフサイクル



- CMKは、実体となる鍵の入れ物(コンテナ)
  - 1つのCMKに対してActiveな鍵は常に1つ
- CDKのライフサイクルは利用者が管理

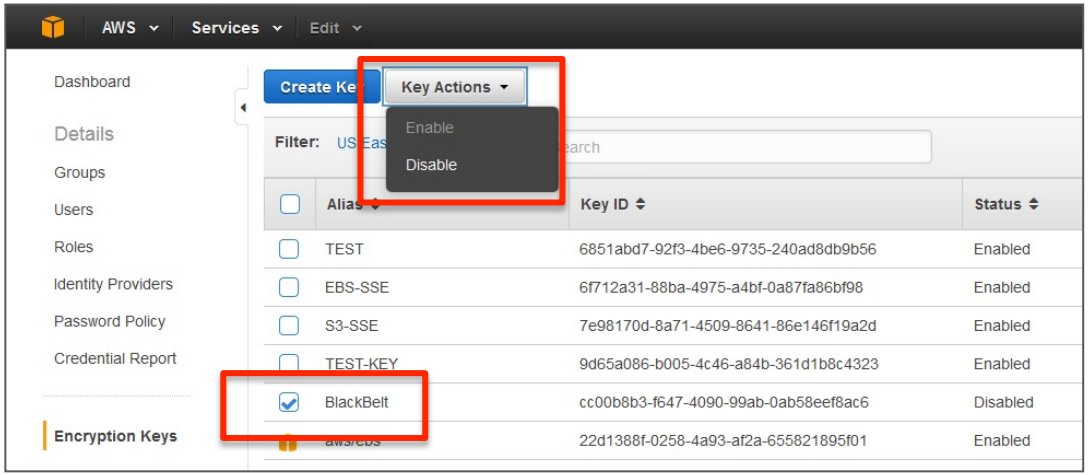
カスタマーマスターキーの生成



- エイリアスは 1～32 文字の長さで、英数字、ダッシュ、スラッシュ（/）、またはアンダースコア
- エイリアスは “aws” で始まってはいけない
- Descriptionは最大 256 文字

- IAMのEncryption Keysより作成
- 暗号鍵は**リージョン単位**で作成
- 暗号鍵に対するアクセスポリシーとして下記に対して管理者及び使用者のパーミッションを設定
  - IAM User
  - IAM Role
  - 外部のAWSアカウント（使用者のみ）
- 設定したKeyポリシーの確認

カスタマーマスターキーのEnable/Disable



- ライフサイクルに関係なく鍵管理者により自由にEnable/Disableが可能
- Disableすると鍵の使用に関する操作が不能
- **Disableされた鍵も課金対象**

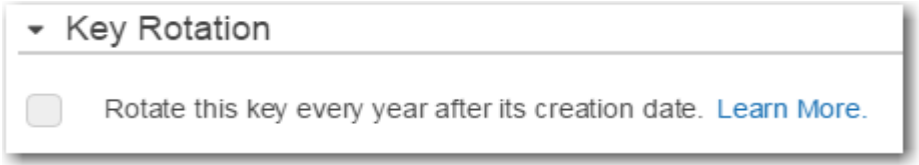
```
$ aws kms generate-data-key --key-id cc00b8b3-f647-4090-99ab-0ab58XXXXXX --key-spec AES_256

A client error (DisabledException) occurred when calling the GenerateDataKey operation: arn:aws:kms:us-east-1:336580663XXX:key/cc00b8b3-f647-4090-99ab-0ab58eeXXXXX is disabled.
```

カスタマーマスターキーの削除

- Disableした鍵を削除する事が可能
  - 7日-30日の待機期間を設定
    - デフォルトは30日
  - 待機期間中であれば削除のキャンセルが可能
- 削除については慎重に！
  - 削除した鍵を利用していたデータは復号不可
  - CloudTrail等を利用して鍵が利用されていないことを確認してから削除

カスタマーマスターキーのローテーション

コンソール (Key Summary Page)

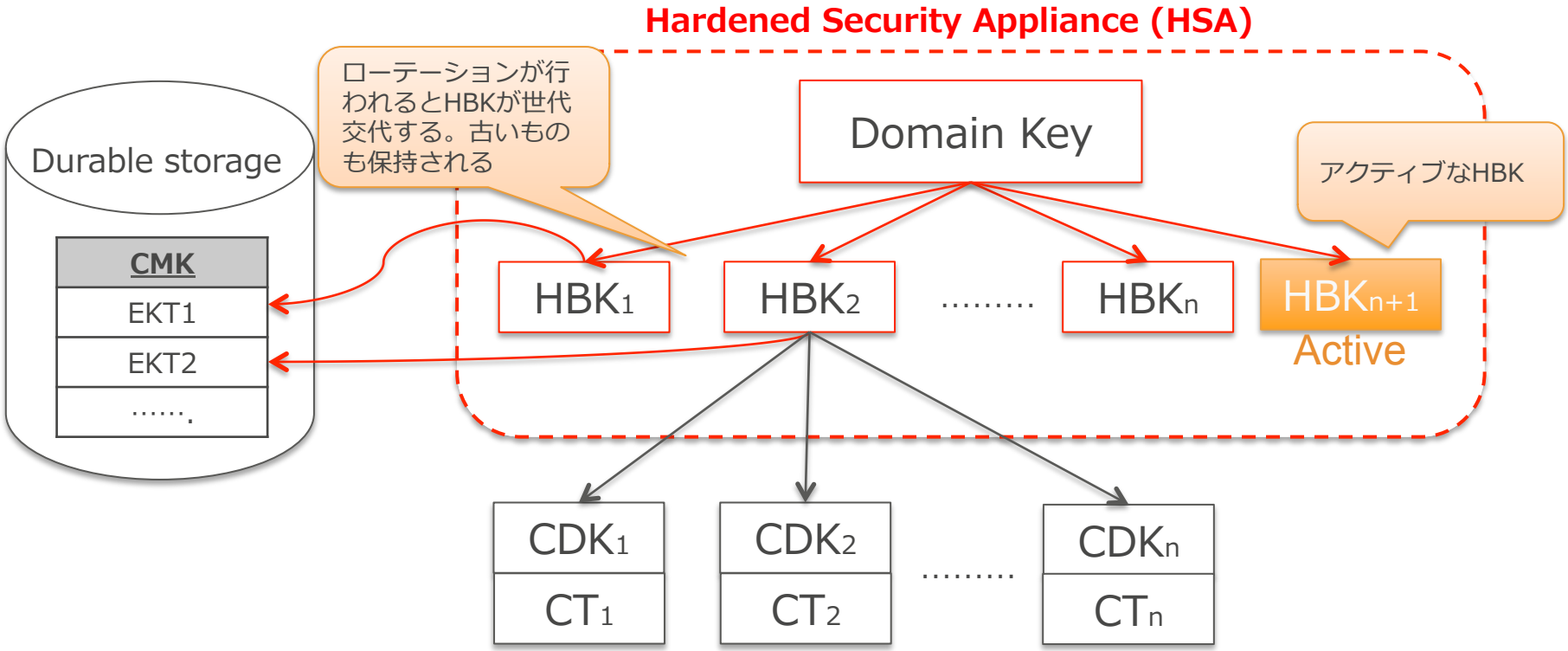


AWS CLI

enable-key-rotation --key-id <value>

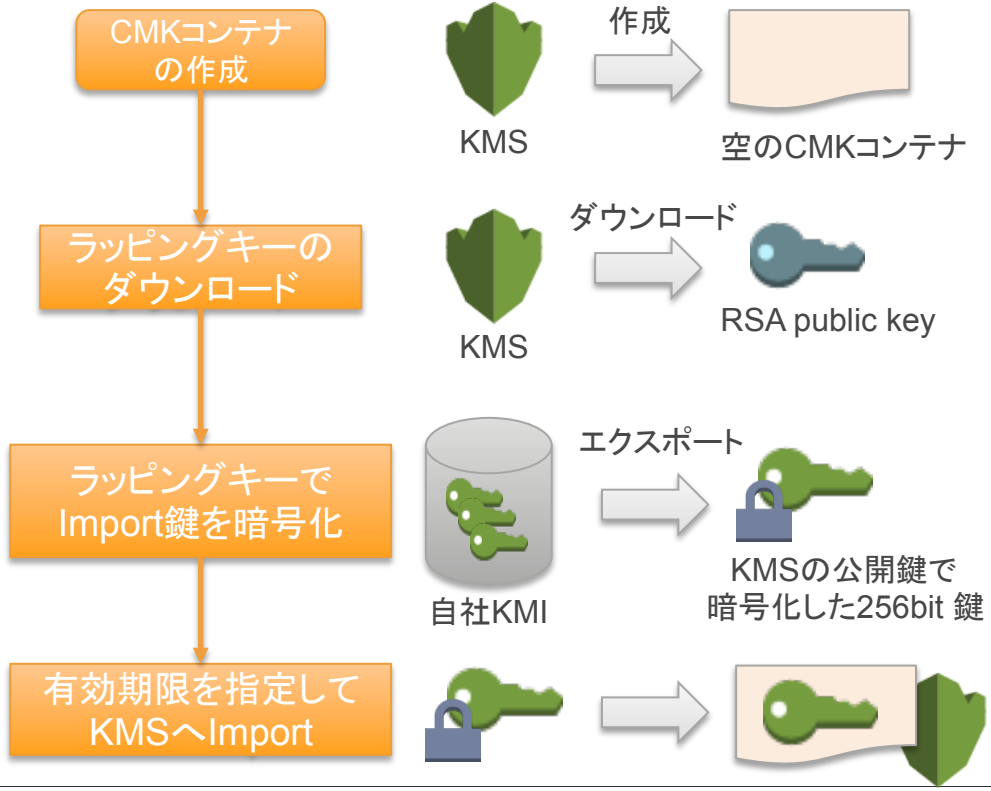
- 一年おきの暗号鍵（CMK）のローテーションを設定可能
  - 有効にした時点から１年後にローテーション
  - 現時点ではローテーション期間は固定
- 古い鍵は保管され続け、復号にのみ利用可能
  - 古いバージョンの鍵も課金対象となるので注意
- 利用者は透過的に新旧の暗号鍵を利用できる
  - 同じKey ID,同じAlias
  - 暗号化リクエストには新しい鍵を利用
  - ReEncrypt APIを利用して明示的に新しい鍵への入れ替えが可能

ローテーションの仕組み



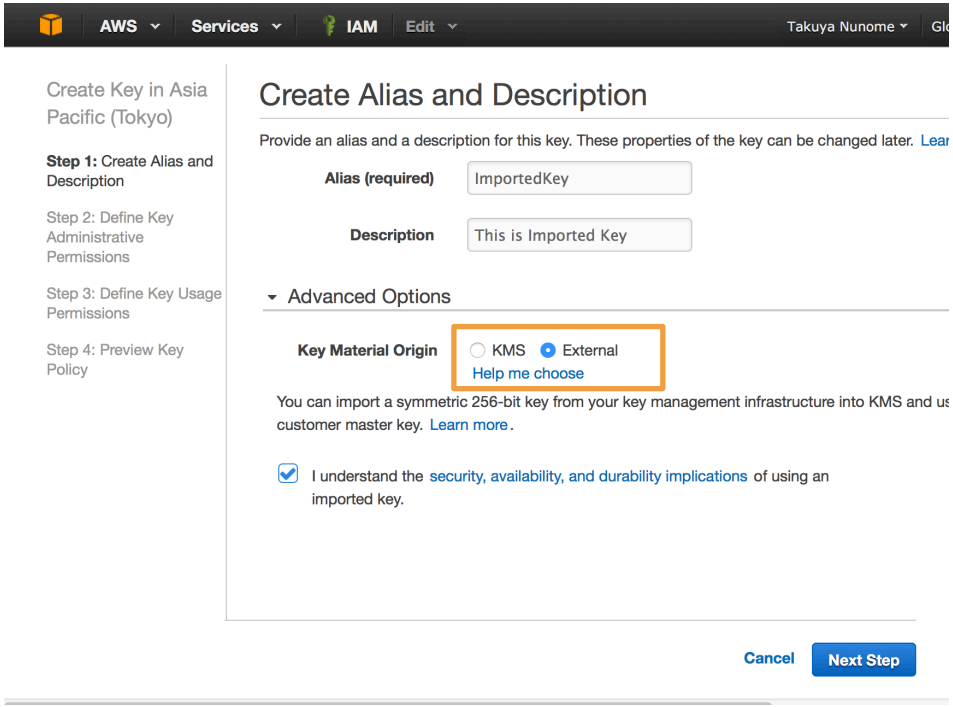
新機能 Bring Your Own Keys New (2016/8/11リリース)

- 鍵のインポートが可能に
  - CMKの生成を利用者がコントロール
  - マスターキーのコピーを自社KMI環境に保持可能
  - 持ち込んだ鍵をKMSとインテグレートされているAWSサービスで利用可能
  - 256bit対称鍵のみサポート
- 鍵の有効期限指定が可能
- 鍵の削除、再インポートが可能



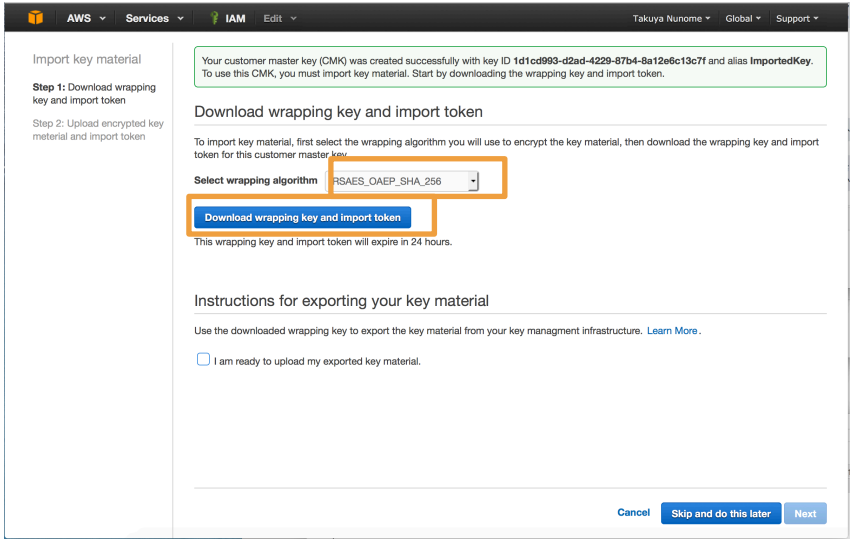
CMKのインポート(1/3)

- CMKコンテナの作成
  - IAMコンソールでImport用に空のCMKコンテナを作成
  - Advanced OptionsでKey Material Originに **External(外部)**を指定
  - 通常のCMK作成と同様に管理者と利用者のIAMユーザーを指定



CMKのインポート(2/3)

- Wrapping keyとImport tokenのダウンロード
  - 自社のKMI環境に合わせてラッピングアルゴリズムを指定
    - RSAES_OAEP_SHA_256(推奨)
    - RSAES_OAEP_SHA1
    - RSAES_PKCS1_V1_5
  - 指定したアルゴリズム用のwrapping keyとimport tokenをダウンロード
    - Import tokenは24時間でExpire
    - CMKコンテナの作成完了後に実施することも可能



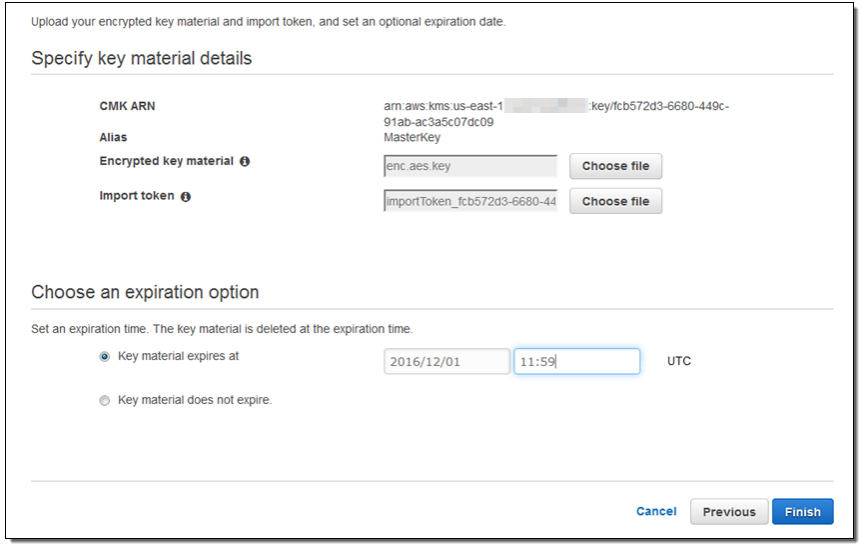
以下の3ファイルが含まれたzipファイルがダウンロードされる
README_<KeyID>_<Timestamp>.txt
importToken_<KeyID>_<Timestamp>
wrappingKey_<KeyID>_<Timestamp>

CMKのインポート(3/3)

- 自身の鍵マテリアルのラッピング
  - 指定したアルゴリズムを利用してラッピング
  - (例)opensslを利用して RSAES_OAEP_SHA_1でラップする場合

```
$ openssl rsautl -encrypt -in plain_text_aes_key.bin -oaep ¥
-inkey wrappingKey_<KEYID>_<TIMESTAMP> ¥
-pubin -keyform DER -out enc.aes.key
```

- ラッピングした鍵とImportToken ファイルを指定して鍵をインポート
  - 有効期限も同時に指定
    - UTCでの日付＆時刻指定または無期限
- 鍵のステータスがPending Import→Enableになれば完了



CMKをインポートする際の注意点

- 鍵のエクスポートは不可
  - 自社KMIでの継続的な鍵管理が必要
- インポートする鍵長は256bitのみ
- 鍵の自動ローテーションは利用不可
- 有効期限が過ぎた鍵はKMSによって削除される
  - CMKにキーマテリアルのImportを再度行えば期限は変更可能
- 有効期限内の鍵削除は通常のCMKと同様
  - 猶予期間(7-30日間)の後CMKごと削除される。
- 同じ鍵を複数リージョンにImportすることは可能
  - 但し、あるリージョンのKMSで暗号化したデータ(データキー)を別リージョンのKMSで復号することは出来ない
    - HBK IDが異なるため

Key Policy

- 暗号鍵に対するリソースベースのパーミッション
- 暗号鍵作成時に定義
- 管理コンソール、CLI/SDKにより変更可能
  - GetKeyPolicy：ポリシーの取得
  - PutKeyPolicy：ポリシーの設定

- 暗号鍵に対するKey Policyの他に、IAM UserやIAM Roleで設定されているPolicyでもアクセス制御が可能

暗号鍵のデフォルトポリシー例

```
{
  "Id": "key-consolepolicy-2",
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Enable IAM User Permissions",
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::336580663xxx:root"
        ]
      },
      "Action": "kms:*",
      "Resource": "*"
    },
    {
      "Sid": "Allow access for Key Administrators",
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::336580663xxx:user/SuperUser"
        ]
      },
```

AWS rootアカウントを許可するdefault policy

```
"Action": [
        "kms:Create*",
        "kms:Describe*",
        "kms:Enable*",
        "kms:List*",
        "kms:Put*",
        "kms:Update*",
        "kms:Revoke*",
        "kms:Disable*",
        "kms:Get*",
        "kms:Delete*"
      ],
      "Resource": "*"
    },
    {
      "Sid": "Allow use of the key",
      "Effect": "Allow",
      "Principal": {
        "AWS": [        "arn:aws:iam::336580663xxx:user/
Dev","arn:aws:iam::336580663xxx:role/
EC2_Admin","arn:aws:iam::912412960xxx:root"
        ]
      },
```

暗号鍵管理者向けポリシー

暗号鍵利用者向けポリシー

暗号鍵のデフォルトポリシー例

```
"Action": [
      "kms:Encrypt",
      "kms:Decrypt",
      "kms:ReEncrypt*",
      "kms:GenerateDataKey*",
      "kms:DescribeKey"
    ],
    "Resource": "*"
  },
  {
    "Sid": "Allow attachment of persistent resources",
    "Effect": "Allow",
    "Principal": {
      "AWS": [
        "arn:aws:iam::33658066XXXX:user/
Dev","arn:aws:iam::33658066XXXX:role/
EC2_Admin","arn:aws:iam::91241296XXXX:root"
      ]
    },
```

暗号鍵利用のためのアクション

```
"Action": [
      "kms:CreateGrant",
      "kms:ListGrants",
      "kms:RevokeGrant"
    ],
    "Resource": "*",
    "Condition": {
      "Bool": {
        "kms:GrantIsForAWSResource": true
      }
    }
  }
]
}
```

KMS と統合された AWS サービスが、キーを使用するための設定

Policyの評価

- 暗号鍵に付与されたリソースベースのパーミッションと、IAM User/Roleに付与したユーザーベースのパーミッションの両方を評価

- 暗号鍵に付与されるDefault Policyはアカウントオーナーに対して全ての権限を持つ

- KMSのアクセス権の評価は通常のIAMのPolicy評価と同じロジック
  - アクセス権限に“Allow”の条件があった場合、アクセス許可
  - ただしアクセス権限に１つでも“Deny”の条件があった場合、アクセス拒否(明示的なDeny)
  - デフォルトDeny < Allow < 明示的なDeny

Grant

- Key Policyと同じくリソースベースのアクセスコントロール機能
- CMKの使用を他のPrincipalに委任
  - AWSサービスからのKMSの利用等
- より細かいアクセス権の設定が可能
- 5つの要素
  - Key　：カスタマーマスターキーのID
  - GranteePrincipal　：パーミッションを受ける対象
  - Operations　：許可されるオペレーション※
  - Constraints　：Grantsが有効になるConditionを定義
  - RetiringPrincipal　：Grantsをretireさせるこのとのできる対象

※許可されるオペレーションにCreateGrantが含まれている場合、
このGrantで許可されたプリンシパルは、制限が同じまたは制限がより厳しいGrantのみ作成可能

Encryption Context

- 暗号化機能を利用する際にKMSに渡すことのできるKey/Valueペア
- 暗号化の際に指定すると復号の際にも同一の値が必要とされる
  - 暗号化されたデータのAdditional Authenticated Data(AAD)として利用可能
- CloudTrailのログに平文で出力されるため、Encryption Contextには機微情報を利用しない
- GrantsのConstraintsの中でも利用
  - EncryptionContextSubset：EncryptionContext中に含まれていればよい
  - EncryptionContextEquals：EncryptionContextが完全に一致した場合
  - EBS暗号化の際に、暗号化を行うvolume IDを利用してGrantsを作成することで、特定ボリュームのみの暗号化を行う

Constraintsの例

- EncryptionContextSubset {“Department”:”Finance”, “classification”:”critcal”}

{“Department”:”Finance”, “classification”:”critical”, “customer”:”12345”} -> OK
{“Department”:” Finance” } -> NG

- EncryptionContextEquals {“Department”:”Finance”, “classification”:”critical”}

{“Department”:”Finance”, “classification”:”critical”} -> OK
{“Department”:” Finance”, “classification”:”critical”, “customer”:”12345”} -> NG

Grantsの利用例

```
"eventName":"CreateGrant",
"awsRegion":"us-east-1",
"sourceIPAddress":"AWS Internal",
"userAgent":"AWS Internal",
"requestParameters":{
        "retiringPrincipal":"137640147550",
        "constraints":{
        "encryptionContextSubset":{
        "aws:ebs:id":"vol-9a98axxx“
        }
        },
        "operations":["Decrypt"],
        "granteePrincipal":"33658066xxxx:aws:ec2-infrastructure:i-5439cxxx",
        "keyId":"arn:aws:kms:us-east-1:33658066xxxx:key/6f712a31-88ba-4975-a4bf-0a87faxxxx“
},
```

EBS暗号化を行う場合
Volumeをアタッチした際のログ

サービスデフォルトキー

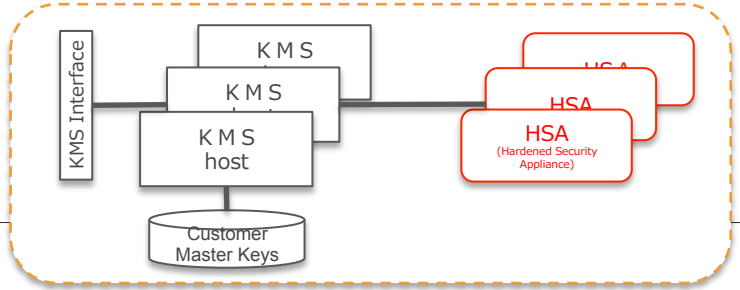
- AWSが管理する各サービスのデフォルトキー
  - エイリアスは “aws/<サービス名>”
- 対応したサービスで暗号化機能を利用する際にアカウント、リージョン毎に自動生成される
  - ex) EBSボリュームの暗号化時
- 3年毎に自動ローテーション
- 各サービスの暗号化に利用できるが、ポリシーの定義等、鍵に対するアクセス権の変更は不可

アジェンダ

- KMSの概要
- KMSの鍵管理
- KMSの暗号化処理
- KMS利用TIPS
- ログ、HSMとの比較、制限、料金等
- まとめ

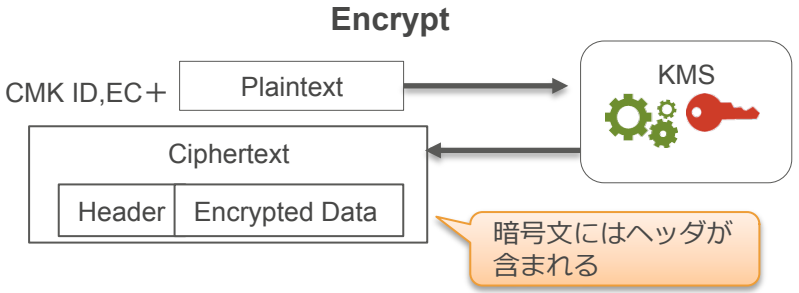
KMSの暗号化処理

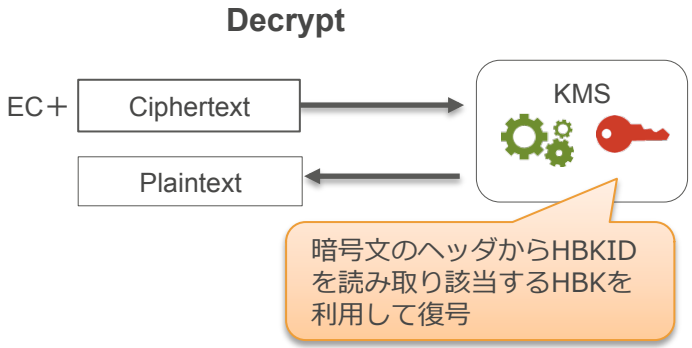
- KMSホスト
  - ユーザーのリクエストを受け、HSA経由で処理
- HSA
  - 鍵の生成、利用、データの暗号化/復号に関わる暗号化機能提供
    - 認証されたリクエスト以外の処理は受け付けない
    - 現時点で暗号化アルゴリズムはAES-GCM 256bitのみ対応
  - 仮想化層を持たない暗号化処理に特化した物理デバイス
- KMS内部の通信は全て暗号化されている



KMSの暗号化と重要なAPI

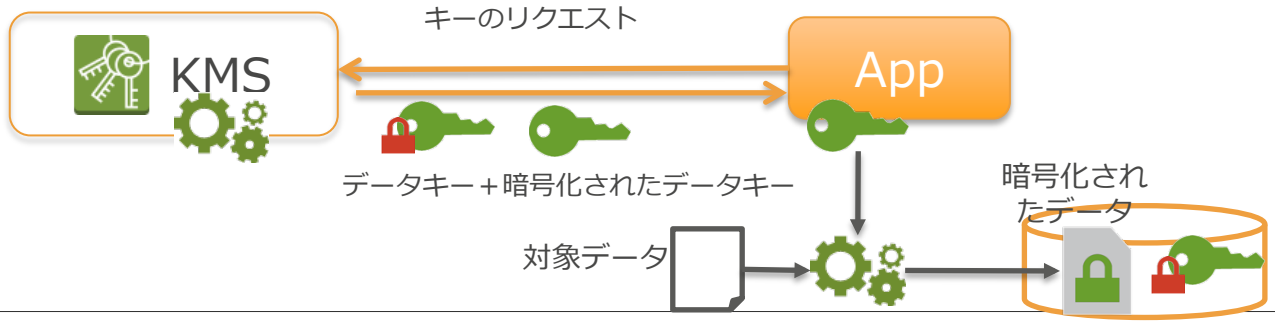
- Encrypt API
  - データを暗号化するためのAPI
    - 鍵データの暗号化用
  - ４KBまでの平文データに対応
  - 生成される暗号文にはヘッダが付与される
    - HBKID、Encryption Contextなどの情報
  - 元データおよび生成された暗号文はAWS内には保持されない
- Decrypt API
  - データを復号するためのAPI
  - CMKの指定は不要
    - 暗号文のヘッダから該当するCMKを特定
  - 暗号文および平文はAWS内には保持されない
- GenerateDataKey
  - ユーザーがデータの暗号化に利用するCDKを生成
  - 平文の鍵と、Encrytpt APIで暗号化された鍵を返す





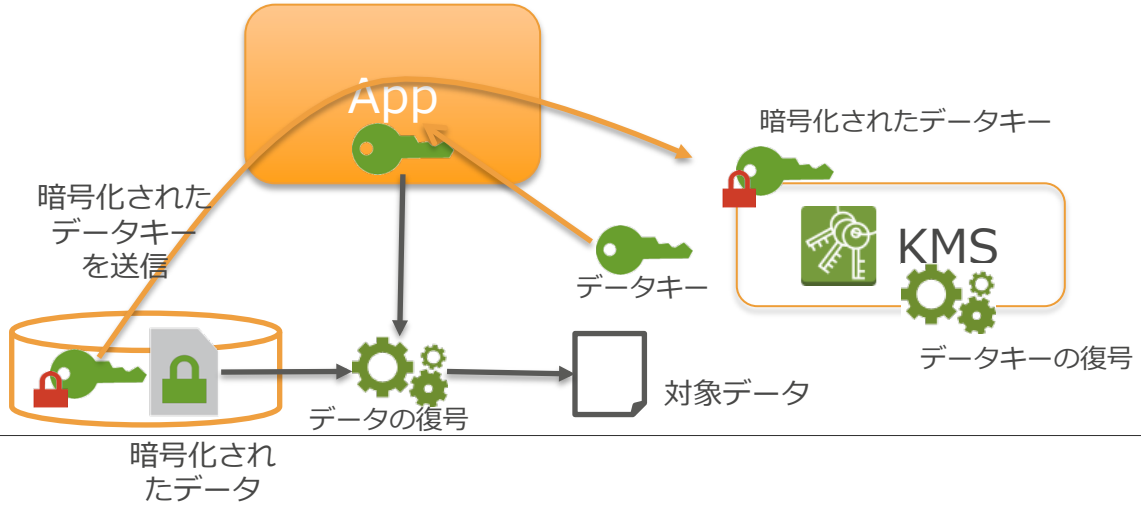
KMSのワークフロー(1/2)

- 暗号化
  - データキーの生成
    - アプリケーションからキーのリクエストがあると、KMSはデータキーと暗号化されたデータキーの両方を返す
  - データキーはアプリケーションのメモリ上に配置し、データの暗号化に利用
    - データキーの管理はアプリケーション側
      - 決してキーをディスクに置かない
      - 暗号化処理が終了したら即座に削除する
  - 暗号化されたデータキーは暗号化されたデータと共に保存する
    - 管理を容易にするため



KMSのワークフロー(2/2)

- 復号
  - アプリケーションから暗号化されたデータキーをKMSに送信
  - KMSは該当するマスターキーを利用してデータキーを復号し、アプリケーションに返す
  - アプリケーションでデータキーを使ってデータを復号
    - データキーの管理はアプリケーション側



KMSを利用したデータ暗号化の選択肢

- Client-side encryption
  - ユーザーアプリケーションでのデータ暗号化にKMSを利用
    - 各種AWS SDKを利用
  - より上位のSDKやクライアントを利用すれば、Envelope Encryptionを容易にハンドル可能
    - AWS Encryption SDK
    - Amazon S3 Client
      - AWS SDK for Java, .NET, Rubyでサポート
    - Amazon EMR File System (EMRFS)
    - Client-side Encryption for Amazon DynamoDB
      - https://github.com/awslabs/aws-dynamodb-encryption-java
- Server-side encryption
  - 各種サービスとインテグレーションされている
  - AWSサービスでデータが受信された後にサービスがKMSを利用してデータを暗号化
  - 利用可能なサービス:
    - S3, Amazon Elastic Block Store (Amazon EBS), Amazon RDS, Amazon Redshift, Amazon WorkMail, Amazon WorkSpaces, AWS CloudTrail, Amazon Simple Email Service (Amazon SES), Amazon Elastic Transcoder, AWS Import/Export Snowball, Amazon Kinesis Firehose, Amazon EMR

AWS SDKを利用したClient-side Encryption

```
//クライアント設定
//USE ROLES FOR EC2
AWSKMSClient kms = new AWSKMSClient();
kms.setRegion("us-west-2");
```

```
//暗号化
String keyId = "alias/my-application-key";
String plaintext = "Hello World!";
Map<String,String> ec = Collections.singletonMap("dept","finance");
GenerateDataKeyRequest req = new GenerateDataKeyRequest()
                                .withEncryptionContext(ec)
                                .withKeyId(keyId)
                                .withKeySpec("AES-256");
DecryptResult res = kms.generateDataKey(req);
ByteBuffer plaintextKey = res.getPlaintext();
ByteBuffer encryptedKey = res.getCiphertextBlob();

//plaintextKeyを利用してデータを暗号化
...
```

KMSのマスターキー

データキーを生成

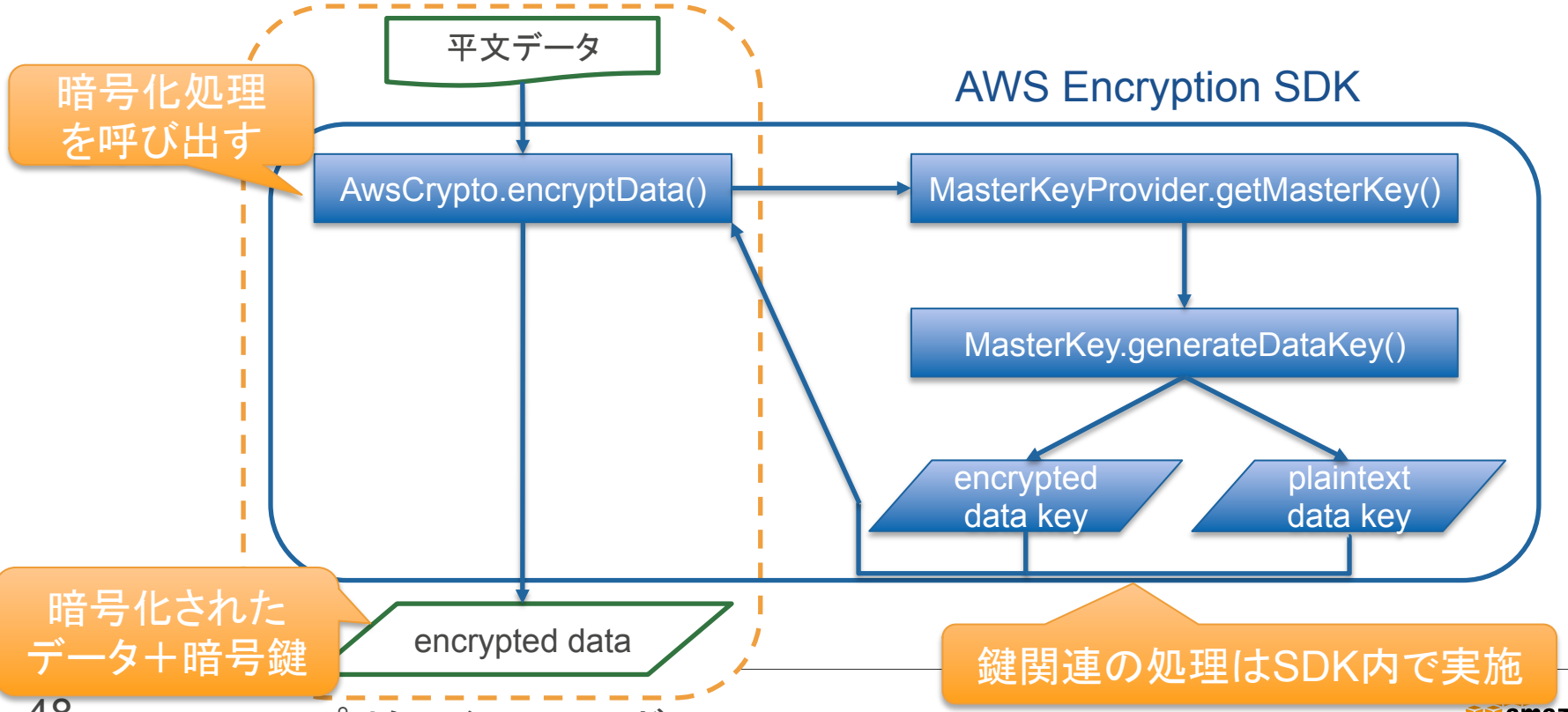
Envelope Encryptionは自分でハンドルする必要がある

AWS Encryption SDK

- AWSが提供するクライアントサイド暗号化のための暗号化ライブラリ
  - ”マスターキープロバイダー”を定義して利用するためのAPIと、トップレベルの鍵もしくはデータを暗号化するための複数の鍵へのインターフェースを提供
  - データ暗号化キー(DEK)の追跡と保護
  - 鍵アクセスのための低レベルの暗号化処理をライブラリ内で実施
- トップレベルのマスターキーを指定すれば、SDKが残りの作業を実施
  - SDKが低レベルの暗号化処理とトップレベルのマスターキーを結びつける
- 鍵プロバイダーを抽象化
  - 複数の鍵を単一のマスターキープロバイダーにまとめて利用
  - 複数の鍵プロバイダーを利用した暗号化(AWS KMSとCloudHSMの同時利用など)
- Javaライブラリとして提供
  - https://github.com/awslabs/aws-encryption-sdk-java

AWS Encryption SDKによる暗号化

鍵プロバイダを利用したEnvelope Encryptionコードを大幅に簡素化



AWS Encryption SDKでKMSを利用したClient-side Encryption

- 暗号化

```
// インスタンス化
final AwsCrypto crypto = new AwsCrypto();
// KmsMasterKeyProviderをセットアップ
final KmsMasterKeyProvider prov = new KmsMasterKeyProvider(keyId);
// 暗号化
final byte[] ciphertext = crypto.encryptData(prov, message);
```

KMS用のMasterKeyProvider

データキーの生成等はSDKが処理
平文のデータキーは暗号化処理終了後にSDKが削除

生成されるciphertextは、暗号化された
データキーと暗号文が結合されている

- 復号

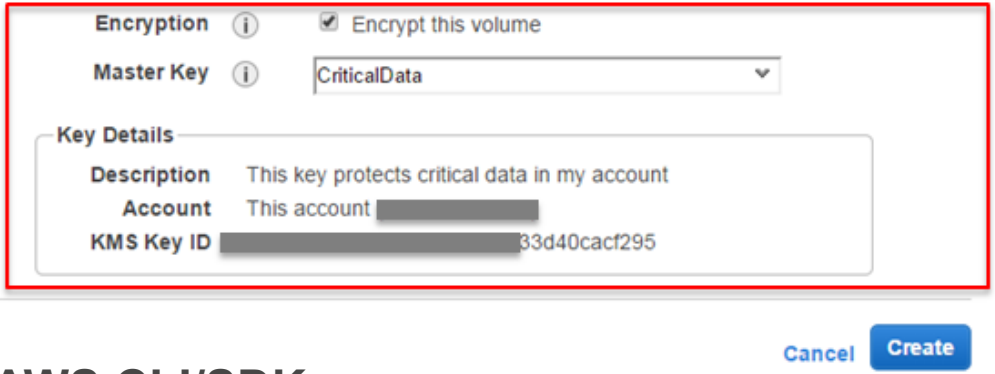
```
final AwsCrypto crypto = new AwsCrypto();
final KmsMasterKeyProvider prov = new KmsMasterKeyProvider(keyId);
// 復号
final CryptoResult<byte[], KmsMasterKey> res = crypto.decryptData(prov, ciphertext);
//  正しい鍵が使われていたかチェック
if (!res.getMasterKeyIds().get(0).equals(keyId)) {
      throw new IllegalStateException("Wrong key id!");
}
byte[] plaintext = res.getResult();
```

KMSとしては復号にはKey IDは不要
だが、データに使われていた鍵が想
定していたものかをチェックしている

AWSサービスでのServer-side Encryption with KMS

EBSの例

Console



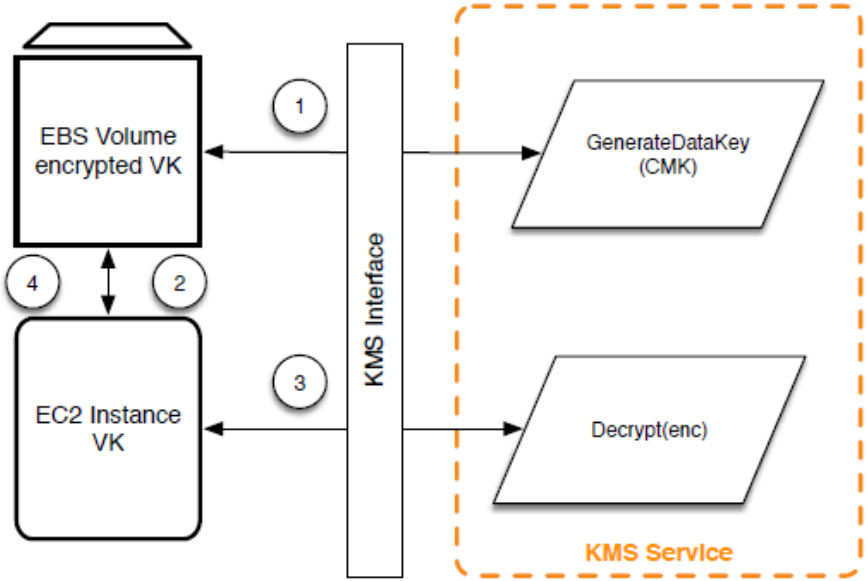
各サービスでCMKを指定するのみ
Envelope Encryptionのハンドルは
サービス側で行われる

AWS CLI/SDK

- create-volume [--dry-run | --no-dry-run] [--size <value>] [--snapshot-id <value>] --availability-zone <value> [--volume-type <value>] [--iops <value>] [--encrypted | --no-encrypted] [--kms-key-id <value>] [--cli-input-json <value>] [--generate-cli-skeleton]

AWSサービスでのServer-Side Encryption with KMS

- EBSの例
  1. EBSがTLSセッションで暗号化されたボリュームキーをKMSから取得、KMSは指定されたキーへのアクセス権をチェックし、暗号化されたデータキーをEBSに送信(EBSは次回のアタッチに備え、暗号化されたデータキーをボリュームのメタデータに保存)
  2. EBSボリュームがマウントされると、EC2がボリュームのメタデータから暗号化されたボリュームキーを取得
  3. KMSへの復号リクエストが生成されEC2インスタンスが稼働するホストが復号されたボリュームキーを受け取る
  4. ボリュームキーはアタッチされたEBSボリュームに対する全ての入出力データの暗号化/復号に利用される



アジェンダ

- KMSの概要
- KMSの鍵管理
- KMSの暗号化処理
- KMS利用TIPS
- ログ、HSMとの比較、制限、料金等
- まとめ

KMSの制限事項

- 各AWSのアカウントごとに以下のリソース制限があります
- 上限緩和にはAWS Support Centerよりチケットをあげてください

リソース	デフォルトの制限
CMK	1000(リージョン毎)
エイリアス	1100(リージョン毎)
CMKあたりのGrant	2500
CMKあたりの特定プリンシパルのGrant	30
1秒あたりのリクエスト	可変(Encrypt/Decrypt/ReEncrypt/ GenerateRandom/GenerateDataKey/ GenerateDataKeyWithoutPlaintextは合計で100)

http://docs.aws.amazon.com/ja_jp/kms/latest/developerguide/limits.html

KMS利用 TIPS

- 直接暗号化/復号できるデータは4KBまで
  - Envelope Encryption方式での利用を推奨
- APIリクエストのスロットリングに注意
  - 暗号化EBSボリュームを複数もつインスタンスの同時起動や、大量オブジェクトのS3へのUpload/Download等に注意
  - 必要に応じて上限緩和申請
    https://docs.aws.amazon.com/ja_jp/kms/latest/developerguide/limits.html
- 対称鍵暗号方式のみサポート
  - 公開鍵暗号方式では利用できない
- リージョン間での鍵の共有は出来ない

KMS with AWSサービス TIPS

- Amazon EBS
  - サポートされているインスタンスタイプに注意
  - ボリュームの生成時にCMKを指定して暗号化
    - 暗号化されていないボリュームの直接の暗号化、暗号化ボリュームの直接の非暗号化は不可
      - 暗号化されていないボリュームのスナップショットをコピーする時に暗号化することは可能
      - 暗号化されたスナップショットをコピーする時に異なるCMKを指定して再暗号化することは可能
  - 暗号化スナップショットを他のAWSアカウントへ共有することが可能
    - CMKに対するアクセス権も付与する必要がある
    - デフォルトキーで暗号化されたスナップショットは共有不可
    - 共有されたスナップショットからボリュームを作成する場合は一旦コピーが必要
  - KMSのCMKで暗号化されたスナップショットはリージョン間コピー不可
  - 1インスタンスにアタッチできる同じCMKで暗号化したボリュームは30個まで

http://docs.aws.amazon.com/ja_jp/AWSEC2/latest/UserGuide/EBSEncryption.html

KMS with AWS サービス TIPS

- Amazon RDS
  - ストレージ暗号化で利用
  - 新規インスタンスの作成時に暗号化指定
  - 暗号化可能なインスタンスタイプに注意
  - 暗号化されたインスタンスの暗号化解除は不可
  - 暗号化されていないスナップショットをコピーして暗号化することは可能(Auroraでは不要)
  - リードレプリカも同じキーを利用して暗号化する必要がある
  - 暗号化されていないバックアップ/スナップショットを暗号化インスタンスにリストアすることは出来ない
    - Auroraでは暗号化されていないスナップショットから暗号化クラスターに復元可能
  - 暗号化されたMySQLスナップショットをAuroraに復元することはできない
  - KMSで暗号化されたスナップショットのリージョン間コピーは不可
  - リージョン間レプリケーションも不可
  - 暗号化スナップショットを他のAWSアカウントへ共有することが可能
    - CMKに対するアクセス権限を付与する必要がある
    - 共有されたスナップショットからインスタンスを作成する場合は一旦コピーが必要
      https://docs.aws.amazon.com/ja_jp/AmazonRDS/latest/UserGuide/USER_ShareSnapshot.html
  - 不用意なキーのDisableは避ける
    - キーをDisableするとTerminalステートになり、リカバリー不能となる
    - バックアップからのリストアが必要になる

http://docs.aws.amazon.com/ja_jp/AmazonRDS/latest/UserGuide/Overview.Encryption.html

KMS with AWS サービス TIPS

- Amazon S3
  - KMSで暗号化されたオブジェクトに対する操作はVersion4署名が必要
  - オブジェクトの暗号化に利用されたデータキーもオブジェクトと共に暗号化されて保管される
  - レスポンスに含まれるETagはオブジェクトのMD5ではない
  - バケットポリシーでSSE-KMSの強制化が可能
    - s3:PutObjectに対して、“s3:x-amz-server-side-encryption”:“aws:kms”の条件を利用
- Amazon SES
  - 受信したメッセージの暗号化にKMSを利用可能
    - S3暗号化クライアントを利用してメッセージをS3に保管
    - Encryption Contextにルール、メッセージIDを指定
  - SESに対してCMKのアクセス権限を付与する必要がある
  - SESには復号権限が無いため、アプリケーションでＳ３からデータを取得して復号する必要がある
    - S3 Clientの利用を推奨
      https://docs.aws.amazon.com/ja_jp/kms/latest/developerguide/services-ses.html

アジェンダ

- KMSの概要
- KMSの鍵管理
- KMSの暗号化処理
- KMS利用TIPS
- ログ、HSMとの比較、制限、料金等
- まとめ

CloudTrailによるKMSのログ集約

- CloudTrailによりKMSの管理コンソールでの操作やSDKでの操作で利用したAPIのログを取得
  - CreateAlias
  - CreateGrant
  - CreateKey
  - Decrypt
  - DeleteAlias
  - DescribeKey
  - DisableKey
  - EnableKey
  - Encrypt
  - GenerateDataKey
  - GenerateDataKeyWithoutPlaintext
  - GenerateRandom
  - GetKeyPolicy
  - ListAliases
  - ListGrants
  - ReEncrypt

CloudTrailにより取得できるログ

- 誰が
- いつ
- どのAPIを
- どのAWSリソースから
- どのAWSリソースに対して

http://docs.aws.amazon.com/ja_jp/kms/latest/developerguide/logging-using-cloudtrail.html

KMSのセキュリティ

- 鍵の管理はお客様が実施
- マスターキーには誰もアクセスできない
- 幅広い堅牢化技術を使用してマスターキーを保護するように設計
  - プレーンテキストのマスターキーをディスクに保存しない
  - メモリ上に存続させ続けない
  - デバイスに接続できるシステムを制限するなど
- サービス内でソフトウェアをアップデートするためのアクセスは複数段階の承認プロセスによって管理
  - Amazonの独立したグループによって監査およびレビューを実施
- キーは作成されたリージョンにのみ保存。他のリージョンに移動できない
- 一年おきのマスターキーの自動ローテーションが可能
- AWS CloudTrailによる管理操作、暗号化操作の記録
- KMSの統制は第三者認証を取得している:
  - Service Organization Control (SOC 1)
  - PCI-DSS

※詳細はAWSコンプライアンスパッケージを参照

AWS KMSとCloud HSMの違い

	AWS CloudHSM	AWS Key Management Service
専有性	VPCにお客様専用のハードウェアデバイス（ Safe Net Luna SA 7000 HSM)をインストール。	マルチテナントのマネージドサービス
可用性	可用性と耐久性はお客様が管理	高可用性、耐久性の高い鍵保管用ストレージと鍵管理
Root of trust	“root of trust”はお客様が管理	“root of trust”はAWSが管理
コンプライアンス	FIPS 140-2 レベル 2及び情報セキュリティ国際評価基準 EAL4+標準に準拠。耐タンパー性を備える。CloudTrailにも対応。	CloudTrailとの統合による監査機能
操作	現在のところ管理コンソール対応無し（CloudHSM CLI等CLIで操作）	管理コンソール、SDK、AWS CLI
サードパーティ製品	EBS用SafeNet ProtectV ボリューム暗号化、Apache、Microsoft SQL Server（透過的データ暗号化）等	（カスタムソフトウェア。AWS SDKは提供）
AWSサービス	Redshift, RDS(Oracle TDE)	S3, EBS, RDS(全エンジン）,Redshift, Elastic Transcoder, WorkMail , EMRFS
暗号化機能	共通及び公開鍵暗号に対応	現在のところ共通鍵暗号のみ
コスト	概ね固定費	従量課金

CloudHSMとKMSの使い分け

- 規制や法令の対応により、認定を受けた鍵管理モジュールが必要
- 現在SafeNetのHSMをオンプレミスで利用している
- 公開鍵暗号化も行いたい
- 暗号化処理をオフロードしたい
- 暗号化リクエストのピークが高い
- 隔離された専用環境で厳格に鍵管理したい

CloudHSM

- 共通鍵暗号のみ利用
- 鍵管理は自分で行いたい
- 手軽かつ安全に暗号化処理を利用したい
- より低コストに鍵管理の仕組みを利用したい
- AWSサービスで簡単に利用したい
- 保管されるデータの暗号化がメイン
- CloudTrailによる監査を行いたい

KMS

KMSの料金

- $1/key version/月
  - KMSで生成、インポートにかかわらない
  - ローテーションを有効にした場合、料金は更新した各バージョンにつき月間$1
  - 課金対象外のkey
    - AWS管理のサービスデフォルトキー
    - 削除を予定されたCMK
      - 待機期間中に削除をキャンセルした場合、削除を予定されなかったものとして課金
    - GenerateDataKey/GenerateDataKeyWithoutPlaintextで生成されたCDK
- $0.03 per 10,000 APIリクエスト (Gov Cloudを除く全てのリージョン)
  - 20,000 req/月の無償利用枠(全リージョン合計)

※価格は2016年9月現在のものです。CloudTrailを有効にしてAPIアクセスログを取得する場合は別途S3,SNSの料金がかかります。

http://aws.amazon.com/jp/kms/pricing/

アジェンダ

- KMSの概要
- KMSの鍵管理
- KMSの暗号化処理
- KMS利用TIPS
- ログ、HSMとの比較、制限、料金等
- まとめ

KMSの特徴

1. S3、EBS等に実装されたAWSによる暗号化（SSE）
2. Userによる暗号鍵の持ち込みによるS3暗号化（CSE）
3. Cloud HSM を用いた暗号化

→ 利用者が暗号化鍵に対するコントロールをもっていない

→ 暗号化の範囲指定、暗号化鍵のローテション等、実行管理が煩雑である

→ Hardwareアプライアンスを用いており高価である

AWS Key Management Serviceを使うことで・・・

1. 暗号鍵はAWS上にSecureに保管、管理はUserにて実施
2. SDKと連携することで、3rd Party製ソフトにも適応可能
3. 比較的安価にKMIを利用可能

まとめ

- AWS Key Management Service は、データを保護するための暗号化キーの一元管理を可能にする低コストなマネージドサービスです

- AWSサービスとの統合により容易にデータの暗号化を行うことができますし、アプリケーション上のデータの暗号化にも利用することができます

- 暗号鍵のセキュアな管理を可能にし、CloudTrailとの統合による組み込み型の監査対応機能も利用できます

参考資料

- AWS Key Management Service Developer Guide
  http://docs.aws.amazon.com/ja_jp/kms/latest/developerguide/overview.html
- AWS Key Management Service API Reference
  http://docs.aws.amazon.com/ja_jp/kms/latest/APIReference/Welcome.html
- AWS Key Management Service FAQ
  http://aws.amazon.com/jp/kms/faqs/
- AWS Key Management Service Pricing
  http://aws.amazon.com/jp/kms/pricing/
- AWS Key Management Service whitepaper
  https://d0.awsstatic.com/whitepapers/KMS-Cryptographic-Details.pdf

オンラインセミナー資料の配置場所

- AWS クラウドサービス活用資料集
  - http://aws.amazon.com/jp/aws-jp-introduction/

- AWS Solutions Architect ブログ
  - 最新の情報、セミナー中のQ&A等が掲載されています
  - http://aws.typepad.com/sajp/

公式Twitter/Facebook
AWSの最新情報をお届けします

@awscloud_jp

もしくは
http://on.fb.me/1vR8yWm

最新技術情報、イベント情報、お役立ち情報、
お得なキャンペーン情報などを日々更新しています！

AWSの導入、お問い合わせのご相談

- AWSクラウド導入に関するご質問、お見積り、資料請求をご希望のお客様は、以下のリンクよりお気軽にご相談ください
  https://aws.amazon.com/jp/contact-us/aws-sales/

※「AWS　問い合わせ」で検索してください